Graph Logics with Rational Relations and the Generalized Intersection Problem

Graph Logics with Rational Relations and the Generalized Intersection Problem

Pablo Barceló
Department of Computer Science
University of Chile
pbarcelo@dcc.uchile.cl

Diego Figueira
School of Informatics
University of Edinburgh
dfigueir@inf.ed.ac.uk

Leonid Libkin
School of Informatics
University of Edinburgh
libkin@inf.ed.ac.uk

***Abstract*—We investigate some basic questions about the interaction of regular and rational relations on words. The primary motivation comes from the study of logics for querying graph topology, which have recently found numerous applications. Such logics use conditions on paths expressed by regular languages and relations, but they often need to be extended by rational relations such as subword (factor) or subsequence. Evaluating formulae in such extended graph logics boils down to checking nonemptiness of the intersection of rational relations with regular or recognizable relations (or, more generally, to the generalized intersection problem, asking whether some projections of a regular relation have a nonempty intersection with a given rational relation).**

We prove that for several basic and commonly used rational relations, the intersection problem with regular relations is either undecidable (e.g., for subword or suffix, and some generalizations), or decidable with non-multiply-recursive complexity (e.g., for subsequence and its generalizations). These results are used to rule out many classes of graph logics that freely combine regular and rational relations, as well as to provide the simplest problem related to verifying lossy channel systems that has non-multiply-recursive complexity. We then prove a dichotomy result for logics combining regular conditions on individual paths and rational relations on paths, by showing that the syntactic form of formulae classifies them into either efficiently checkable or undecidable cases. We also give examples of rational relations for which such logics are decidable even without syntactic restrictions.

I. Introduction

The motivation for the problems investigated in this paper comes from the study of logics for querying graphs. Such logics form the basis of query languages for graph databases, that have recently found numerous applications in areas including biological networks, social networks, Semantic Web, crime detection, etc. (see [1] for a survey) and led to multiple systems and prototypes. In such applications, data is usually represented as a labeled graph. For instance, in social networks, people are nodes, and labeled edges represent different types of relationship between them; in RDF – the underlying data model of the Semantic Web – data is modeled as a graph, with RDF triples naturally representing labeled edges.

The questions that we address are related to the interaction of various classes of relations on words, for instance, rational relations (examples of those include subword and subsequence) or regular relations (such as prefix, or equality of words). An example of a question we are interested in is as follows: is it decidable whether a given a regular relation contains a pair (w, w') so that w is a subword/subsequence of w'? Problems like this are very basic and deserve a study on their own right, but they are also necessary to answer questions on the power and complexity of querying graph databases. We now explain how they arise in that setting.

Logical languages for querying graph data have been developed since the late 1980s (and some of them became precursors of languages later used for XML). They query the topology of the graph, often leaving querying data that might be stored in the nodes to a standard database engine. Such logics are quite different in their nature and applications from another class of graph logics based on spatial calculi [10], [17]. Their formulae combine reachability patterns. The simplest form is known as *regular path queries (RPQs)* [16], [15]; they check the existence of a path whose label belongs to a regular language. Those are typically used as atoms and then closed under conjunction and existential quantification, resulting in the class of *conjunctive regular path queries (*CRPQ*s)*, which have been the subject of much investigation [8], [18], [21]. For instance, a CRPQ may ask for nodes v so that there exist nodes v_1 and v_2 and paths from v to v_i with the label in a regular language L_i.

The expressiveness of these queries, however, became insufficient in applications such as the Semantic Web or biological networks due to their inability to *compare* paths. For instance, it is a common requirement in RDF languages to compare paths based on specific semantic associations [2]; biological sequences often need to be compared for similarity, based, for example, on the edit distance.

To address this, an extension of CRPQs with relations on paths was proposed [3]. It used *regular* relations on paths, i.e., relations given by synchronized automata [20], [22]. Equivalently, these are the relations definable in automatic structures on words [4], [6], [7]. They include prefix, equality, equal length of words, or fixed edit distance between words. The extension of CRPQs with them, called ECRPQs, was shown to have acceptable complexity (NLogSpace with respect to data, PSpace with respect to query).

However, it was still short of the expressiveness needed in many applications. For instance, semantic associations between paths used in RDF applications often deal with subwords or subsequences, but these relations are *not* regular.

They are *rational*: they are still accepted by automata, but those whose heads move asynchronously. Adding them to a query language must be done with extreme care: simply replacing regular relations with rational in the definition of ECRPQs makes query evaluation undecidable!

So we set out to investigate the following problem: given a class of graph queries, e.g., CRPQs or ECRPQs, what happens if one adds the ability to test whether pairs of paths belong to a rational relation S, such as subword or subsequence? We start by observing that this problem is a generalization of the *intersection problem*: given a regular relation R, and a rational relation S, is $R \cap S \neq \emptyset$? It is well known that there exist rational relations S for which it is undecidable [5]; however, we are not interested in artificial relations obtained by encoding PCP instances, but rather in very concrete relations used in querying graph data.

The intersection problem captures the essence of graph logics ECRPQs and CRPQs (for the latter, when restricted to the class of recognizable relations [5], [14]). In fact, query evaluation can be cast as the *generalized intersection problem*. Its input includes an m-ary regular relation R, a binary rational relation S, and a set I of pairs from $\{1, \ldots, m\}$. It asks whether there is a tuple $(w_1, \ldots, w_m) \in R$ so that $(w_i, w_j) \in S$ whenever $(i, j) \in I$. For $m = 2$ and $I = \{(1, 2)\}$, this is the usual intersection problem.

Another motivation for looking at these basic problems comes from verification of lossy channel systems (finite-state processes that communicate over unbounded, but lossy, FIFO channels). Their reachability problem is known to be decidable, although the complexity is not bounded by any multiply-recursive function [13]. In fact, a "canonical" problem used in reductions showing this enormous complexity [12], [13] can be restated as follows: given a binary rational relation R, does it have a pair (w, w') so that w is a subsequence of w'? This naturally leads to the question whether the same bounds hold for the simpler instance of the intersection problem when we use regular relations instead of rational ones. We actually show that this is true.

Summary of results: We start by showing that evaluating CRPQs and ECRPQs extended with a rational relation S can be cast as the generalized intersection problem for S with recognizable and regular relations respectively. Moreover, the complexity of the basic intersection problem is a lower bound for the complexity of query evaluation.

We then study the complexity of the intersection problem for fixed relations S. For recognizable relations, it is well known to be efficiently decidable for every rational S. For regular relations, we show that if S is the subword, or the suffix relation, then the problem is undecidable. That is, it is undecidable to check, given a binary regular relation R, whether it contains a pair (w, w') so that w is a subword of w', or even a suffix of w'. We also present a generalization of this result.

The analogous problem for the subsequence relation is known to be decidable, and, if the input is a rational relation R, then the complexity is non-multiply-recursive [12]. We extend this in two ways. First, we show that the lower bound remains true even for regular relations R. Second, we extend decidability to the class of all rational relations for which one projection is closed under subsequence (the subsequence relation itself is trivially such, obtained by closing the first projection of the equality relation).

In addition to establishing some basic facts about classes of relations on words, these results tell us about the infeasibility of adding rational relations to ECRPQs: in fact adding subword makes query evaluation undecidable, and while it remains decidable with subsequence, the complexity is prohibitively high.

So we then turn to the generalized intersection problem with recognizable relations, corresponding to the evaluation of CRPQs with an extra relation S. We show that the shape of the relation I holds the key to decidability. If its underlying undirected graph is acyclic, then the problem is decidable in PSPACE for every rational relation S (and for a fixed formula the complexity drops to NLOGSPACE). In the cyclic case, the problem is undecidable for some rational relation S. For relations generalizing subsequence, we have decidability when I is a DAG, and for subsequence itself, as well as for suffix, query evaluation is decidable regardless of the shape of CRPQs.

Thus, under the mild syntactic restriction of acyclicity of comparisons with respect to rational relations, such relations can be added to the common class CRPQ of graph queries, without incurring a high complexity cost.

Organization: We give basic definitions in Section II and define the main problems we study in Section III. Section IV introduces graph logics and establishes their connection with the (generalized) intersection problem. Section V studies decidable and undecidable cases of the intersection problem. Section VI looks at the case of recognizable relations and CRPQs and establishes decidability results based on the intersection pattern. Complete proofs of all results are available in the full version of the paper.

II. PRELIMINARIES

Alphabets, languages, and morphisms: We shall use letters Σ, Γ to denote finite alphabets. The set of all finite words over an alphabet Σ is denoted by Σ^*. We write ε for the empty word, $w \cdot w'$ for the concatenation of two words, and $|w|$ for the length of a word w.

If $w = w' \cdot u \cdot w''$, then

- u is a *subword* of w (also called *factor* in the literature, written as $u \preceq w$),
- w' is a *prefix* of w (written as $w' \preceq_{\text{pref}} w$), and
- w'' is a *suffix* of w (written as $w'' \preceq_{\text{suff}} w$).

We say that w' is a *subsequence* of w (also called *subword embedding* in the literature, written as $w' \sqsubseteq w$) if w' is obtained by removing some letters (perhaps none) from w, i.e., $w = a_1 \ldots a_n$, and $w' = a_{i_1} a_{i_2} \ldots a_{i_k}$, where $1 \leq i_1 < i_2 < \ldots < i_k \leq n$.

Recall that a *monoid* $M = \langle U, \cdot, 1\rangle$ has an associative binary operation $\cdot$ and a neutral element 1 satisfying $1x = x1 = x$ for all x (we often write xy for $x \cdot y$). The set Σ^* with the operation of concatenation and the neutral element ε forms a monoid $\langle \Sigma^*, \cdot, \varepsilon\rangle$, the free monoid generated by Σ. A function $f : M \to M'$ between two monoids is a *morphism* if it sends the neutral element of M to the neutral element of M', and if $f(xy) = f(x)f(y)$ for all $x, y \in M$. Every morphism $f : \langle \Sigma^*, \cdot, \varepsilon\rangle \to M$ is uniquely determined by the values $f(a)$, for $a \in \Sigma$, as $f(a_1 \dots a_n) = f(a_1) \cdot \dots \cdot f(a_n)$. A morphism $f : \langle \Sigma^*, \cdot, \varepsilon\rangle \to \langle \Gamma^*, \cdot, \varepsilon\rangle$ is called *alphabetic* if $f(a) \in \Gamma \cup \{\varepsilon\}$, and *strictly alphabetic* if $f(a) \in \Gamma$ for each $a \in \Sigma$, see [5].

A language L is a subset of Σ^*. It is *recognizable* if there is a finite monoid M, a morphism $f : \langle \Sigma^*, \cdot, \varepsilon\rangle \to M$, and a subset M_0 of M such that $L = f^{-1}(M_0)$.

A language L is *regular* if there exists an NFA (nondeterministic finite automaton) $\mathcal{A} = \langle Q, \Sigma, q_0, \delta, F\rangle$ such that $L = \mathcal{L}(\mathcal{A})$, the language of words accepted by $\mathcal{A}$. We use the standard notation for NFAs, where Q is the set of states, q_0 is the initial state, F is the set of final states, and $\delta \subseteq Q \times \Sigma \times Q$ is the transition relation.

A language is *rational* if it is denoted by a regular expression; such expressions are built from $\emptyset$, ε, and alphabet letters by using operations of concatenation ($e \cdot e'$), union ($e \cup e'$), and Kleene star (e^*). It is of course the classical result of formal language theory that the classes of recognizable, regular, and rational languages coincide.

Recognizable, regular, and rational relations: While the notions of recognizability, regularity, and rationality coincide over languages $L \subseteq \Sigma^*$, they differ over relations over Σ, i.e., subsets of $\Sigma^* \times \dots \times \Sigma^*$. We now define those (see [5], [11], [14], [20], [22], [30]).

Since $\langle \Sigma^*, \cdot, \varepsilon\rangle$ is a monoid, the product $(\Sigma^*)^n$ has the structure of a monoid too. We can thus define *recognizable n-ary relations* over Σ as subsets $R \subseteq (\Sigma^*)^n$ so that there exists a finite monoid M and a morphism $f : (\Sigma^*)^n \to M$ such that $R = f^{-1}(M_0)$ for some $M_0 \subseteq M$. The class of n-ary recognizable relations will be denoted by REC_n; when n is clear or irrelevant, we write just REC.

It is well-known that a relation $R \subseteq (\Sigma^*)^n$ is in REC_n iff it is a finite union of the sets of the form $L_1 \times \dots \times L_n$, where each L_i is a regular language over Σ, see [5], [20].

Next, we define the class of regular relations. Let $\perp \notin \Sigma$ be a new alphabet letter, and let $\Sigma_\perp$ be $\Sigma \cup \{\perp\}$. Each tuple $\bar{w} = (w_1, \dots, w_n)$ of words from Σ^* can be viewed as a word over $\Sigma_\perp^n$ as follows: pad words w_i with $\perp$ so that they all are of the same length, and use as the kth symbol of the new word the n-tuple of the kth symbols of the padded words. Formally, let $\ell = \max_i |w_i|$. Then $w_1 \otimes \dots \otimes w_n$ is a word of length ℓ whose kth symbol is $(a_1, \dots, a_n) \in \Sigma_\perp^n$ such that

$$a_i = \begin{cases} \text{the } k\text{th letter of } w_i & \text{if } |w_i| \geq k \\ \perp & \text{otherwise.} \end{cases}$$

We shall also write $\otimes\bar{w}$ for $w_1 \otimes \dots \otimes w_n$. A relation $R \subseteq (\Sigma^*)^n$ is called a *regular n-ary relation* over Σ if there is a finite automaton $\mathcal{A}$ over $\Sigma_\perp^n$ that accepts $\{\otimes\bar{w} \mid \bar{w} \in R\}$. The class of n-ary regular relations is denoted by REG_n; as before, we write REG when n is clear or irrelevant.

Finally, we define rational relations. There are two equivalent ways of doing it. One uses regular expressions, which are now built from tuples $\bar{a} \in (\Sigma \cup \{\varepsilon\})^n$ using the same operations of union, concatenation, and Kleene star. Binary relations $\preceq_{\text{suff}}$, $\preceq$, and $\sqsubseteq$ are all rational: the expression $\big(\bigcup_{a\in\Sigma}(\varepsilon, a)\big)^* \cdot \big(\bigcup_{a\in\Sigma}(a, a)\big)^*$ defines $\preceq_{\text{suff}}$, the expression $\big(\bigcup_{a\in\Sigma}(\varepsilon, a)\big)^* \cdot \big(\bigcup_{a\in\Sigma}(a, a)\big)^* \cdot \big(\bigcup_{a\in\Sigma}(\varepsilon, a)\big)^*$ defines $\preceq$, and the expression $\big(\bigcup_{a\in\Sigma}(\varepsilon, a) \cup (a, a)\big)^*$ defines $\sqsubseteq$.

Alternatively, n-ary rational relations can be defined by means of n-tape automata, that have n heads for the tapes and one additional control; at every step, based on the state and the letters it is reading, the automaton can enter a new state and move some (but not necessarily all) tape heads. The classes of n-ary relations so defined are called *rational n-ary relations*; we use the notation RAT_n or just RAT, as before.

Relationships between classes of relations: While it very well known that $\mathsf{REC}_1 = \mathsf{REG}_1 = \mathsf{RAT}_1$, we have strict inclusions

$$\mathsf{REC}_k \subsetneq \mathsf{REG}_k \subsetneq \mathsf{RAT}_k$$

for every $k > 1$. For instance, $\preceq_{\text{pref}} \in \mathsf{REG}_2 - \mathsf{REC}_2$ and $\preceq_{\text{suff}} \in \mathsf{RAT}_2 - \mathsf{REG}_2$.

The classes of recognizable and regular relations are closed under intersection; however the class of rational relations is not. In fact one can find $R \in \mathsf{REG}_2$ and $S \in \mathsf{RAT}_2$ so that $R \cap S \notin \mathsf{RAT}_2$. However, if $R \in \mathsf{REC}_m$ and $S \in \mathsf{RAT}_m$, then $R \cap S \in \mathsf{RAT}_m$.

Binary rational relations can be characterized as follows [5], [28]. A relation $R \subseteq \Sigma^* \times \Sigma^*$ is rational iff there is a finite alphabet Γ, a regular language $L \subseteq \Gamma^*$ and two alphabetic morphisms $f, g : \Gamma^* \to \Sigma^*$ such that $R = \{(f(w), g(w)) \mid w \in L\}$. If we require f and g to be strictly alphabetic morphisms, we get the class of *length-preserving* regular relations, i.e., $R \in \mathsf{REG}_2$ so that $(w, w') \in R$ implies $|w| = |w'|$. Regular binary relations are then finite unions of relations of the form $\{(w \cdot u, w') \mid (w, w') \in R,\ u \in L\}$ and $\{(w, w' \cdot u) \mid (w, w') \in R,\ u \in L\}$, where R ranges over length-preserving regular relations, and L over regular languages.

Properties of classes of relations: Since relations in REC and REG are given by NFAs, they inherit all the closure/decidability properties of regular languages. If $R \in \mathsf{RAT}$, then each of its projections is a regular language, and can be effectively constructed (e.g., from the description of R as an n-tape automaton). Hence, the nonemptiness problem is decidable for rational relations. However, testing nonemptiness of the intersection of two rational relations is undecidable [5]. Also, for $R, R' \in \mathsf{RAT}$, the following are undecidable: checking whether $R \subseteq R'$ or $R = R'$, universality ($R = \Sigma^* \times \Sigma^*$), and checking whether $R \in \mathsf{REG}$ or $R \in \mathsf{REC}$ [5], [11], [26].

Remark: We defined recognizable, regular, and rational relations over the same alphabet, i.e., as subsets of $(\Sigma^*)^n$. Of course it is possible to define them as subsets of $\Sigma_1 \times \ldots \times \Sigma_n$, with the Σ_i's not necessarily distinct. Technically, there are no differences and all the results will continue to hold. Indeed, one can simply consider a new alphabet Σ as the disjoint union of Σ_i's, and enforce the condition that the ith projection only use the letters from Σ_i (this is possible for all the classes of relations we consider). In fact, in the proofs we shall be using both types of relations.

III. Generalized intersection problem

We now formalize the main technical problem we study. Let $\mathcal{R}$ be a class of relations over Σ, and $\mathcal{S}$ a class of binary relations over Σ. We use the notation $[m]$ for $\{1, \ldots, m\}$. The *generalized intersection problem* $(\mathcal{R} \cap_I \mathcal{S}) \stackrel{?}{=} \emptyset$ is defined as:

PROBLEM:	$(\mathcal{R} \cap_I \mathcal{S}) \stackrel{?}{=} \emptyset$
INPUT:	an m-ary relation $R \in \mathcal{R}$, a relation $S \in \mathcal{S}$, and $I \subseteq [m]^2$
QUESTION:	is there $\bar{w} = (w_1, \ldots, w_m) \in R$ so that $(w_i, w_j) \in S$ for all $(i,j) \in I$?

If $\mathcal{S} = \{S\}$, we write S instead of $\{S\}$. We write $\text{GENINT}_S(\mathcal{R})$ for the class of all problems $(\mathcal{R} \cap_I S) \stackrel{?}{=} \emptyset$ where S is fixed, i.e., the input consists of $R \in \mathcal{R}$ and I. As was explained in the introduction, this problem captures the essence of evaluating queries in various graph logics, e.g., CRPQs or ECRPQs extended with rational relations S. The classes $\mathcal{R}$ will typically be REC and REG.

If $m = 2$ and $I = \{(1,2)\}$, the generalized intersection problem becomes simply the *intersection problem* for the classes $\mathcal{R}$ and $\mathcal{S}$ of binary relations:

PROBLEM:	$(\mathcal{R} \cap \mathcal{S}) \stackrel{?}{=} \emptyset$
INPUT:	$R \in \mathcal{R}$ and $S \in \mathcal{S}$
QUESTION:	is $R \cap S \neq \emptyset$?

The problem $(\mathsf{REC} \cap S) \stackrel{?}{=} \emptyset$ is decidable for every rational relation S, simply by constructing $R \cap S$, which is a rational relation, and testing its nonemptiness. However, $(\mathsf{REG} \cap S) \stackrel{?}{=} \emptyset$ could already be undecidable (we shall give one particularly simple example later).

IV. Graph logics and the generalized intersection problem

In this section we show how the (generalized) intersection problems provide us with upper and lower bounds on the complexity of evaluating a variety of logical queries over graphs. We start by recalling the basic classes of logics used in querying graph data, and show that extending them with rational relations allows us to cast the query evaluation problem as an instance of the generalized intersection problem. The key observations are that:

- the complexity of $\text{GENINT}_S(\mathsf{REC})$ and $(\mathsf{REC} \cap S) \stackrel{?}{=} \emptyset$ provide an upper and a lower bound for the complexity of evaluating CRPQ(S) queries; and
- for ECRPQ(S), these bounds are provided by the complexity of $\text{GENINT}_S(\mathsf{REG})$ and of $(\mathsf{REG} \cap S) \stackrel{?}{=} \emptyset$.

The standard abstraction of graph databases [1] is finite Σ-labeled graphs $G = \langle V, E \rangle$, where V is a finite set of nodes, or vertices, and $E \subseteq V \times \Sigma \times V$ is a set of labeled edges. A *path* ρ from v_0 to v_m in G is a sequence of edges (v_0, a_0, v_1), $(v_1, a_1, v_2), \cdots, (v_{m-1}, a_{m-1}, v_m)$ from E, for some $m \geq 0$. The *label* of ρ, denoted by $\lambda(\rho)$, is the word $a_0 \cdots a_{m-1} \in \Sigma^*$.

The main building blocks for graph queries are *regular path queries*, or *RPQ*s [16]; they are expressions of the form $x \xrightarrow{L} y$, where L is a regular language. We normally assume that L is represented by a regular expression or an NFA. Given a Σ-labeled graph $G = \langle V, E \rangle$, the answer to an RPQ above is the set of pairs of nodes (v, v') such that there is a path ρ from v to v' with $\lambda(\rho) \in L$.

Conjunctive RPQs, or CRPQs [8], [9], [15] are the closure of RPQs under conjunction and existential quantification. Formally, they are expressions of the form

$$\varphi(\bar{x}) \quad = \quad \exists \bar{y} \bigwedge_{i=1}^{m} (u_i \xrightarrow{L_i} u'_i) \tag{1}$$

where variables u_i, u'_is come from $\bar{x}, \bar{y}$. The semantics naturally extends the semantics of RPQs: $\varphi(\bar{a})$ is true in G iff there is a tuple $\bar{b}$ of nodes such that for every $i \leq m$ and every v_i, v'_i interpreting u_i and u'_i, respectively, we have a path ρ_i between v_i and v'_i whose label $\lambda(\rho_i)$ is in L_i.

CRPQs can further be extended to *compare* paths. For that, we need to name path variables, and choose a class of allowed relations on paths. The simplest such extension is the class of CRPQ(S) queries, where S is a binary relation over Σ^*. Its formulae are of the form

$$\varphi(\bar{x}) \quad = \quad \exists \bar{y} \left(\bigwedge_{i=1}^{m} (u_i \xrightarrow{\chi_i : L_i} u'_i) \quad \wedge \quad \bigwedge_{(i,j) \in I} S(\chi_i, \chi_j) \right) \tag{2}$$

where $I \subseteq [m]^2$. We use variables $\chi_1, \ldots, \chi_m$ to denote paths; these are quantified existentially. That is, the semantics of $G \models \varphi(\bar{a})$ is that there is a tuple $\bar{b}$ of nodes and paths ρ_k, for $k \leq m$, between v_k and v'_k (where, as before, v_k, v'_k are elements of $\bar{a}, \bar{b}$ interpreting u_k, u'_k) such that $(\lambda(\rho_i), \lambda(\rho_j)) \in S$ whenever $(i,j) \in I$. For instance, the query

$$\exists y, y' \ \left((x \xrightarrow{\chi : \Sigma^* a} y) \wedge (x \xrightarrow{\chi' : \Sigma^* b} y') \wedge \chi \sqsubseteq \chi' \right)$$

finds nodes v so that there are two paths starting from v, one ending with an a-edge, whose label is a subsequence of the other one, that ends with a b-edge.

The input to the *query evaluation problem* consists of a graph G, a tuple $\bar{v}$ of nodes, and a query $\varphi(\bar{x})$; the question is whether $G \models \varphi(\bar{v})$. This corresponds to the *combined complexity* of query evaluation. In the context of query evaluation, one is often interested in *data complexity*, when the typically small formula φ is fixed, and the input consists of the typically large graph $(G, \bar{v})$. We now relate it to the complexity of $\text{GENINT}_S(\mathsf{REC})$.

Lemma IV.1. *Fix a* CRPQ*(S) query φ as in (2). Then there is a* DLOGSPACE *algorithm that, given a graph G and a tuple $\bar{v}$ of nodes, constructs an m-ary relation $R \in \mathsf{REC}$ so that the answer to the generalized intersection problem $(R \cap_I S) \stackrel{?}{=} \emptyset$ is 'yes' iff $G \models \varphi(\bar{v})$.*

Proof idea. Given a Σ-labeled graph $G = \langle V, E\rangle$ and two nodes v, v', we write $\mathcal{A}(G, v, v')$ for G viewed as an NFA with the initial state v and the final state v'. Now consider a CRPQ(S) query $\varphi(\bar{x})$ given by (2). Let $\bar{v}$ be a tuple of nodes of G, of the same length as $\bar{x}$.

The algorithm first enumerates all tuples $\bar{b}$ of nodes of G of the same length as $\bar{y}$. Let n_i and n'_i be the interpretations of u_i and u'_i, when $\bar{x}$ is interpreted as $\bar{v}$ and $\bar{y}$ as $\bar{b}$. Define $R_{\bar{b}} = \prod_{i=1}^{m}(\mathcal{L}(\mathcal{A}(G, n_i, n'_i)) \cap L_i)$; this is a relation in REC_m. Hence, $R = \bigcup_{\bar{b}} R_{\bar{b}}$ in REC_m too. It is now easy to see that $R \cap_I S \neq \emptyset$ iff $G \models \varphi(\bar{v})$. □

Conversely, the intersection problem for recognizable relations and S can be encoded as answering CRPQ(S) queries.

Lemma IV.2. *For each binary relation S, there is a* CRPQ*(S) query $\varphi(x, x')$ and a* DLOGSPACE *algorithm that, given a relation $R \in \mathsf{REC}_2$, constructs a graph G and two nodes v, v' so that $G \models \varphi(v, v')$ iff $R \cap S \neq \emptyset$.*

Proof idea. Let $R \in \mathsf{REC}_2$ be given as $\bigcup_{i=1}^{n}(L_i \times K_i)$, where the $L_i, K_i \subseteq \Sigma^*$ are regular languages for every i. Let $\langle V_i, E_i\rangle$ be the underlying graph of the NFA defining L_i, such that v_0^i is the initial state, and F_i is the set of final states. Likewise we define $\langle W_i, H_i\rangle$, nodes w_0^i and sets $C_i \subseteq W_i$ for NFA defining K_i.

We now construct the graph G. Its labeling alphabet is the union of Σ and $\{\#, \$, !\}$. Its set of vertices is the disjoint union of all the V_is, W_is, as well as two distinguished nodes *start* and *end*. Its edges include all the edges from E_is and H_is, and the following:

- $\#$-labeled edges from *start* to each initial state, i.e., to each v_i^0 and w_i^0 for all $i \leq n$.
- \$-labeled edges between the initial states of automata with the same index, i.e., edges $(v_0^i, \$, w_0^i)$ for all $i \leq n$.
- !-labeled edges from final states to *end*, i.e., edges $(v, !, \textit{end})$, where $v \in \bigcup_{i\leq n} F_i \cup \bigcup_{i\leq n} C_i$.

The CRPQ(S) query $\varphi(x, y)$ is given below; it omits path variables for paths that are not used in comparisons:

$$\exists x_1, x_2, z_1, z_2 \left(\begin{array}{cccc} & x \xrightarrow{\#} x_1 & \wedge & x \xrightarrow{\#} x_2 \\ \wedge & x_1 \xrightarrow{\chi:\Sigma^*} z_1 & \wedge & x_2 \xrightarrow{\chi':\Sigma^*} z_2 \\ \wedge & z_1 \xrightarrow{!} y & \wedge & z_2 \xrightarrow{!} y \\ \wedge & x_1 \xrightarrow{\$} x_2 & \wedge & S(\chi, \chi') \end{array}\right)$$

It is routine to verify that $G \models \varphi(\textit{start}, \textit{end})$ iff $R \cap S \neq \emptyset$. □

Combining the lemmas, we obtain:

Theorem IV.3. *Let $\mathcal{K}$ be a complexity class closed under* DLOGSPACE *reductions. Then:*

1) *If the problem* $\textsc{GenInt}_S(\mathsf{REC})$ *is in $\mathcal{K}$, then data complexity of* CRPQ*(S) queries is in $\mathcal{K}$; and*
2) *If the problem $(\mathsf{REC} \cap S) \stackrel{?}{=} \emptyset$ is hard for $\mathcal{K}$, then so is data complexity of* CRPQ*(S) queries.*

We now consider *extended CRPQs*, or *ECRPQs*, which enhance CRPQs with regular relations [3], and prove a similar result for them, with the role of REC now played by REG. Formally, ECRPQs are expressions of the form

$$\varphi(\bar{x}) \quad = \quad \exists \bar{y}\, \Big(\bigwedge_{i=1}^{m} (u_i \xrightarrow{\chi_i:L_i} u'_i) \quad \wedge \quad \bigwedge_{j=1}^{k} R_j(\bar{\chi}_j) \Big) \qquad (3)$$

where each R_j is a relation from REG, and $\bar{\chi}_j$ a tuple from $\chi_1, \ldots, \chi_m$ of the same arity as R_j. The semantics of course extends the semantics of CRPQs: the witnessing paths $\rho_1, \ldots, \rho_m$ should also satisfy the condition that for every atom $R(\rho_{i_1}, \ldots, \rho_{i_l})$ in (3), the tuple $(\lambda(\rho_{i_1}), \ldots, \lambda(\rho_{i_l}))$ is in R.

Finally, we obtain ECRPQ(S) queries by adding comparisons with respect to a relation $S \in \mathsf{RAT}$, getting a class of queries $\varphi(\bar{x})$ of the form

$$\exists \bar{y}\, \Big(\bigwedge_{i=1}^{m} (u_i \xrightarrow{\chi_i:L_i} u'_i) \wedge \bigwedge_{j=1}^{k} R_j(\bar{\chi}_j) \wedge \bigwedge_{(i,j)\in I} S(\chi_i, \chi_j) \Big) \qquad (4)$$

Exact analogs of Lemmas IV.1 and IV.2 hold, with ECRPQs replacing CRPQs and REG replacing REC. Hence, we get:

Theorem IV.4. *Let $\mathcal{K}$ be a complexity class closed under* DLOGSPACE *reductions. Then:*

1) *If the problem* $\textsc{GenInt}_S(\mathsf{REG})$ *is in $\mathcal{K}$, then data complexity of* ECRPQ*(S) queries is in $\mathcal{K}$; and*
2) *If the problem $(\mathsf{REG} \cap S) \stackrel{?}{=} \emptyset$ is hard for $\mathcal{K}$, then so is data complexity of* ECRPQ*(S) queries.*

Thus, our next goal is to understand the behaviors of the generalized intersection problem for various rational relations S which are of interest in graph logics; those include subword, suffix, subsequence. In fact to rule out many undecidable or infeasible cases it is often sufficient to analyze the intersection problem. We do this in the next section, and then analyze the decidable cases to come up with graph logics that can be extended with rational relations.

V. THE INTERSECTION PROBLEM: DECIDABLE AND UNDECIDABLE CASES

We now study the problem $(\mathsf{REG} \cap S) \stackrel{?}{=} \emptyset$ for binary rational relations S such as subword and subsequence, and for classes of relations generalizing them. The input is a binary regular relation R over Σ, given by an NFA over $\Sigma_\perp \times \Sigma_\perp$. The question is whether $R \cap S \neq \emptyset$. We also derive results about the complexity of ECRPQ(S) queries. For all lower-bound results in this section, we assume that the alphabet contains at least two symbols.

As already mentioned, there exist rational relations S such that $(\mathsf{REG} \cap S) \stackrel{?}{=} \emptyset$ is undecidable. However, we are interested in relations that are useful in graph querying, and that are among the most commonly used rational relations, and for them the status of the problem was unknown.

Note that the problem $(\mathsf{REC} \cap S) \stackrel{?}{=} \emptyset$ is tractable: given $R \in \mathsf{REC}$, the relation $R \cap S$ is rational, can be efficiently constructed, and checked for nonemptiness.

A. Undecidable cases: subword and relatives

We now show that even for such simple relations as subword and suffix, the intersection problem is undecidable. That is, given an NFA over $\Sigma_\perp \times \Sigma_\perp$ defining a regular relation R, the problem of checking for the existence of a pair $(w, w') \in R$ with $w \preceq_{\mathrm{suff}} w'$ or $w \preceq w'$ is undecidable.

Theorem V.1. *The problems* $(\mathsf{REG} \cap \preceq_{\mathrm{suff}}) \stackrel{?}{=} \emptyset$ *and* $(\mathsf{REG} \cap \preceq) \stackrel{?}{=} \emptyset$ *are undecidable.*

As an immediate consequence of this, we obtain:

Corollary V.2. *The query evaluation problem for* ECRPQ($\preceq_{\mathrm{suff}}$) *and* ECRPQ($\preceq$) *is undecidable.*

Thus, some of the most commonly used rational relations cannot be added to ECRPQs without imposing further restrictions.

Proof idea. We sketch the idea of the proof for $\preceq_{\mathrm{suff}}$. We encode nonemptiness for linearly bounded automata (LBA). The alphabet Σ is the disjoint union of the tape alphabet of the LBA, its states, and the designated symbol \$. Each configuration C with the tape content $a_0 \dots a_n$, where a_0 and a_n are the left and right markers, the state is q, and the head points at a_i is encoded as a word $w_C = \$a_0 \dots a_{i-1} q a_i \dots a_n\$$. Note that the relation $\{(w_C, w_{C'}) \mid C' \text{ is an immediate successor of } C\}$ is regular and hence so is the relation $R = \{(w_{C_0} w_{C_1} \dots w_{C_m}, w_{C'_1} \dots w_{C'_m}) \mid C'_{i+1}$ is an immediate successor of C_i for $i < m\}$, since all configuration encodings are of the same length. In fact, taking product with a regular language, we can also assume that R enforces C_0 to be an initial configuration, and C_m to be a final configuration. If $R \cap \preceq_{\mathrm{suff}}$ is nonempty, it contains a pair $(w_{C_0} w_{C_1} \dots w_{C_m}, w_{C_1} \dots w_{C_m})$ such that C_{i+1} is an immediate successor of C_i for all $i < m$, i.e., iff there is an accepting computation of the LBA. This proves undecidability. The proof for $\preceq$ is very similar. □

Note that the relation R constructed in the proof is definable in first-order logic, so the intersection problem for suffix and subword is undecidable even if the input relation comes from the class of star-free regular relations.

The essence of the undecidability result is that relations such as $\preceq_{\mathrm{suff}}$ and $\preceq$ can be decomposed in a way that one of the components of the decomposition is a graph of a nontrivial strictly alphabetic morphism. More precisely, let $R \cdot R'$ be the binary relation $\{(w \cdot w', u \cdot u') \mid (w, u) \in R \text{ and } (w', u') \in R'\}$. Let $\mathrm{Graph}(f)$ be the graph of a function $f : \Sigma^* \to \Sigma^*$, i.e., $\{(w, f(w)) \mid w \in \Sigma^*\}$.

Proposition V.3. *Let* R_0, R_1 *be binary relations on* Σ *such that* R_0 *is recognizable and its second projection is* Σ^*. *Let* f *be a strictly alphabetic morphism that is not constant. Then, for* $S = R_0 \cdot \mathrm{Graph}(f) \cdot R_1$, *the problem* $(\mathsf{REG} \cap S) \stackrel{?}{=} \emptyset$ *is undecidable.*

Note that both $\preceq_{\mathrm{suff}}$ and $\preceq$ are of the required shape: suffix is $(\{\varepsilon\} \times \Sigma^*) \cdot \mathrm{Graph(id)} \cdot (\{\varepsilon\} \times \{\varepsilon\})$, and subword is $(\{\varepsilon\} \times \Sigma^*) \cdot \mathrm{Graph(id)} \cdot (\{\varepsilon\} \times \Sigma^*)$, where id is the identity alphabetic morphism.

B. Decidable cases: subsequence and relatives

We now show that the intersection problem is decidable for the subsequence relation $\sqsubseteq$ and, much more generally, for a class of relations that do not, like the relations considered in the previous section, have a "rigid" part. More precisely, for relations one of whose projections is closed under taking subsequences, we also retain decidability. However, the complexity bounds are extremely high. In fact we show that the complexity of checking whether $(R \cap \sqsubseteq) \neq \emptyset$, when R ranges over REG_2, is not bounded by any multiply-recursive function. This was previously known for R ranging over RAT_2, and was viewed as the simplest problem with non-multiply-recursive complexity [12]. We now push it further and show that this high complexity is already achieved with regular relations.

Some of the ideas for showing this come from a decidable relaxation of the Post Correspondence Problem (PCP), namely the *regular Post Embedding Problem*, or PEP^{reg}, introduced in [12], and shown to be in the level $\mathrm{F}_{\omega^\omega}$ of the fast-growing hierarchy of recursive functions [27], [29]. The input to the problem consists of two morphisms $f, g : \Sigma^* \to \Gamma^*$ and a regular language $L \subseteq \Sigma^*$; it asks whether there is some $w \in L$ such that $f(w) \sqsubseteq g(w)$ (recall that in the case of the PCP the question is whether $f(w) = g(w)$ with $L = \Sigma^+$). This problem is known to be decidable, and as hard as the reachability problem for lossy channel systems [12] which cannot be bounded by any primitive-recursive function —in fact, by any multiple-recursive function [29].

The problem PEP^{reg} is just a reformulation of the problem $(\mathsf{RAT} \cap \sqsubseteq) \stackrel{?}{=} \emptyset$. Indeed, relations of the form $\{(f(w), g(w)) \mid w \in L\}$, where $L \subseteq \Sigma^*$ ranges over regular languages and f, g over morphisms $\Sigma^* \to \Gamma^*$ are precisely the relations in RAT_2 [5], [28]). Hence, $(\mathsf{RAT} \cap \sqsubseteq) \stackrel{?}{=} \emptyset$ is decidable, with non-multiply-recursive complexity. We show that the lower bound already applies to regular relations.

Theorem V.4. *The problem* $(\mathsf{REG} \cap \sqsubseteq) \stackrel{?}{=} \emptyset$ *is decidable, and its complexity is not bounded by any multiply-recursive function.*

Proof idea. As already mentioned, decidability follows from [12]. To prove the lower bound, we first show that the existence of a solution to PEP^{reg} is equivalent to the existence of a special solution, which we call a *strict codirect solution*.[1] A word $w = a_1 \cdots a_m \in \Sigma^*$ is a strict codirect solution if $f(a_1 \cdots a_m) \sqsubseteq g(a_1 \cdots a_m)$ and for every $i < m$, $f(a_1 \cdots a_i) \not\sqsubseteq g(a_1 \cdots a_i)$. We then show how to code

[1] This is a slightly more restrictive definition than the *codirect solutions* used for the decidability of PEP^{reg} in [12], which is essential to make possible our reduction from the $(\mathsf{REG} \cap \sqsubseteq) \stackrel{?}{=} \emptyset$ problem.

the existence of a strict codirect solution as an instance of $(\mathsf{REG} \cap \sqsubseteq) \stackrel{?}{=} \emptyset$.

Given a rational relation $R \subseteq \Sigma^* \times \Gamma^*$ we convert it into a length-preserving regular relation $R' \subseteq \Sigma_\perp^* \times \Gamma_\perp^*$ so that R is the set of elements of R' projected onto $\Sigma^* \times \Gamma^*$, and if $(q, (a, b), q')$ is a transition of the NFA accepting R' so is $(q, (\perp, \perp), q')$. If we now let R'' to be the regular relation $R' \cdot \{(\epsilon, v) \mid v \in \{\perp\}^*\}$, we obtain that:

(i) if $w \in R'' \cap \sqsubseteq$ then $w' \in R \cap \sqsubseteq$, where w' is the projection of w onto $\Sigma^* \times \Gamma^*$; and
(ii) for any $w' \in R \cap \sqsubseteq$ there is some strict codirect solution $w \in R'' \cap \sqsubseteq$ such that w' is the projection of w onto $\Sigma^* \times \Gamma^*$.

Thus the theorem follows.

Whereas (i) is trivial, (ii) follows from the fact that w is a strict codirect solution. If $w' = (u, v) \in R''$, where $f(w) = (u)_\Gamma$, $g(w) = (v)_\Gamma$, the complication is now that, since $u \in \Sigma_\perp$, it could be that $u \not\sqsubseteq v$ just because there is some $\perp$ in u that does not appear in v. But we build (u, v) such that whenever $u[i] = \perp$ forces $v[j] = \perp$ with $j > i$ then we also have that $u[j] = \perp$. This repeats, forcing $v[k] = \perp$ for some $k > j$ and so on, until we reach the tail of v that has sufficiently many $\perp$'s to satisfy all the accumulated demands for occurrences of $\perp$. □

Note that one cannot solve the problem $(\mathsf{REG} \cap \sqsubseteq) \stackrel{?}{=} \emptyset$ by reducing to nonemptiness of rational relations due to the following.

Proposition V.5. *There is a binary regular relation R such that $(R \cap \sqsubseteq)$ is not rational.*

The next question is how far we can extend the decidability of $(\mathsf{RAT} \cap \sqsubseteq) \stackrel{?}{=} \emptyset$. It turns out that if we allow one projection of a rational relation to be closed under taking subsequences, then we retain decidability.

Let $R \subseteq \Sigma^* \times \Sigma^*$ be a binary relation. Define another binary relation

$$R_\sqsubseteq = \{(u, w) \mid u \sqsubseteq u' \text{ and } (u', w) \in R \text{ for some } u'\}$$

Then the class of *subsequence-closed relations*, or SCR, is the class $\{R_\sqsubseteq \mid R \in \mathsf{RAT}\}$. Note that the subsequence relation itself is in SCR, since it is obtained by closing the (regular) equality relation under subsequence. That is, $\sqsubseteq \; = \; \{(w, w) \mid w \in \Sigma^*\}_\sqsubseteq$. Not all rational relations are subsequence-closed (for instance, subword is not).

The following summarizes properties of subsequence-closed relations.

Proposition V.6.

1) $\mathsf{SCR} \subsetneq \mathsf{RAT}$.
2) $\mathsf{SCR} \not\subseteq \mathsf{REG}$ *and* $\mathsf{REG} \not\subseteq \mathsf{SCR}$.
3) A relation R is in SCR *iff $\{w \otimes w' \mid (w, w') \in R\}$ is accepted by an NFA $\mathcal{A} = \langle Q, \Sigma_\perp \times \Sigma_\perp, q_0, \delta, F\rangle$ such that $(q, (a, b), q') \in \delta$ implies $(q, (\perp, b), q') \in \delta$ for all $q, q' \in Q$ and $a, b \in \Sigma_\perp$.*

When an SCR relation is given as an input to a problem, we assume that it is represented as an NFA in item 3 in the above proposition.

Note also that $(\mathsf{SCR} \cap \mathsf{SCR}) \stackrel{?}{=} \emptyset$ is decidable in polynomial time: if $R, R' \in \mathsf{SCR}$ and $R \cap R' \neq \emptyset$, then $(\varepsilon, w) \in R \cap R'$ for some w, and hence the problem reduces to simple NFA nonemptiness checking.

The main result about SCR relations generalizes decidability of $(\mathsf{RAT} \cap \sqsubseteq) \stackrel{?}{=} \emptyset$.

Theorem V.7. *The problem $(\mathsf{RAT} \cap \mathsf{SCR}) \stackrel{?}{=} \emptyset$ is decidable.*

Of course the complexity is non-multiply-recursive, since this subsumes $(\mathsf{REG} \cap \sqsubseteq) \stackrel{?}{=} \emptyset$ of Theorem V.4.

Proof idea. Decidability is shown by reduction to a problem where solutions have a specific shape. Given a rational binary relation R_0 and a subsequence-closed relation R_1 defined by two automata $\mathcal{A}_0$ and $\mathcal{A}_1$ over $\Sigma_\perp \times \Sigma_\perp$, we say that (w_0, w_1) is a *solution* if $w_0 = u_0 \otimes v_0 \in \mathcal{L}(\mathcal{A}_0)$, $w_1 = u_1 \otimes v_1 \in \mathcal{L}(\mathcal{A}_1)$ and $(u_0)_\Sigma = (u_1)_\Sigma$, $(v_0)_\Sigma = (v_1)_\Sigma$, where w_Σ denotes projection of w onto Σ. We say that (w_0, w_1) is a *synchronized solution* if it further satisfies $v_0 = v_1$.

The problem of finding a solution reduces to that of finding a synchronized solution. Indeed, consider the automata $\mathcal{A}_0', \mathcal{A}_1'$ as the result of adding all transitions $(q, (\perp, \perp), q)$ for every possible state q to both automata. It is clear that the relations recognized by these automata remain unchanged, and that $\mathcal{A}_0'$ is still a SCR automaton. It is easy to verify that there is a synchronized solution for $(\mathcal{A}_0', \mathcal{A}_1')$ if, and only if, there is a solution for $(\mathcal{A}_0, \mathcal{A}_1)$.

The problem of finding a synchronized solution for $\mathcal{A}_0, \mathcal{A}_1$ can be then formulated as the problem of finding words $v, u_0, u_1 \in \Sigma_\perp^*$ with $|v| = |u_0| = |u_1|$, so that $(u_0 \otimes v, u_1 \otimes v)$ is a solution. We can compute an automaton $\mathcal{A}$ over $\Sigma_\perp^3$ from $\mathcal{A}_0, \mathcal{A}_1$, such that $(u_0, u_1, v) \in \mathcal{L}(\mathcal{A})$ if, and only if, $u_0 \otimes v \in \mathcal{L}(\mathcal{A}_0)$ and $u_1 \otimes v \in \mathcal{L}(\mathcal{A}_1)$. Consider now an automaton $\mathcal{A}'$ over $\Sigma_\perp^2$ such that $\mathcal{L}(\mathcal{A}') = \{(u_0, u_1) \mid \exists v \; (u_0, u_1, v) \in \mathcal{L}(\mathcal{A})\}$. It corresponds to the rational automaton of the projection onto the first and second components of the ternary relation of $\mathcal{A}$, and it can be computed from $\mathcal{A}$ in polynomial time. We then deduce that there exists $u_0 \otimes u_1 \in \mathcal{L}(\mathcal{A}')$ so that $(u_0)_\Sigma \sqsubseteq (u_1)_\Sigma$ if, and only if, there is $v \in \Sigma_\perp^*$ with $|v| = |u_0| = |u_1|$ so that $u_0 \otimes v \in \mathcal{L}(\mathcal{A}_0)$ and $u_1 \otimes v \in \mathcal{L}(\mathcal{A}_1)$, where $(u_0)_\Sigma \sqsubseteq (u_1)_\Sigma$. But this is equivalent to $R_0 \cap R_1 \neq \emptyset$, since

- if $((u_1)_\Sigma, (v)_\Sigma) \in R_1$ and $(u_0)_\Sigma \sqsubseteq (u_1)_\Sigma$, then $((u_0)_\Sigma, (v)_\Sigma) \in R_1$ (since $R_1 \in \mathsf{SCR}$) and hence $((u_0)_\Sigma, (v)_\Sigma) \in R_0 \cap R_1$; and
- if $R_0 \cap R_1 \neq \emptyset$, then there exists a synchronized solution $(u_0 \otimes v, u_1 \otimes v)$ of $\mathcal{A}_0, \mathcal{A}_1$ (where $u_0 \otimes v \in \mathcal{L}(\mathcal{A}_0)$, and $u_1 \otimes v \in \mathcal{L}(\mathcal{A}_1)$, and $(u_0)_\Sigma = (u_1)_\Sigma$).

We have thus reduced the problem to $(\mathsf{RAT} \cap \sqsubseteq) \stackrel{?}{=} \emptyset$, which is decidable [12], as already mentioned. □

Coming back to graph logics, we obtain:

Corollary V.8. *The complexity of evaluation of* ECRPQ($\sqsubseteq$) *queries is not bounded by a multiply-recursive function.*

Another corollary can be stated in purely language-theoretic terms.

Corollary V.9. *Let $\mathcal{C}$ be a class of binary relations on Σ^* that is closed under intersection and contains* REG*. Then the nonemptiness problem for $\mathcal{C}$ is:*

- *undecidable if $\preceq$ or $\preceq_{\text{suff}}$ is in $\mathcal{C}$;*
- *non-multiply-recursive if $\sqsubseteq$ is in $\mathcal{C}$.*

Discussion: In addition to answering some basic language-theoretic questions about the interaction of regular and rational relations, and to providing the simplest yet problem with non-multiply-recursive complexity, our results also ruled out logical languages for graph databases that freely combine regular relations and some of the most commonly used rational relations, such as subword and subsequence. With them, query evaluation becomes either undecidable or non-multiply-recursive (which means that no realistic algorithm will be able to solve the hard instances of this problem).

This does not yet fully answer our questions about the evaluation of queries in graph logics. First, in the case of subsequence (or, more generally, SCR relations) we still do not know if query evaluation of ECRPQs with such relations is decidable (i.e., what happens with $\text{GENINT}_S(\text{REG})$ for such relations S).

Even more importantly, we do not yet know what happens with the complexity of CRPQs (i.e., $\text{GENINT}_S(\text{REC})$ for various relations S. These questions are answered in the next section.

VI. Restricted logics and generalized intersection problem

The previous section already ruled out some graph logics with rational relations as either undecidable or decidable with extremely high complexity. This was done merely by analyzing the intersection problem for binary rational and regular relations. We now move to the study of the generalized intersection problem, and use it to analyze the complexity of graph logics in full generality. We first deal with the generalization of the decidable case (SCR relations), and then consider the problem $\text{GENINT}_S(\text{REC})$, corresponding to CRPQs extended with relations S on paths.

A. Generalized intersection problem and subsequence

We know that $(\text{REG} \cap \sqsubseteq) \stackrel{?}{=} \emptyset$ is decidable, although not multiply-recursive. What about its generalized version? It turns out it remains decidable.

Theorem VI.1. *The problem $\text{GENINT}_{\sqsubseteq}(\text{REG})$ is decidable. That is, there is an algorithm that decides, for a given m-ary regular relation R and $I \subseteq [m]^2$, whether $R \cap_I \sqsubseteq \neq \emptyset$.*

For checking decidability we explore the solution space tree as in the proof of Theorem V.7. However, this time the notion of saturation is different, since we need to consider a different condition for each component of the m-ary relation. The finiteness of the computed tree follows from Higman's Lemma this time in combination with Dickson's Lemma [19].

Corollary VI.2. *The query evaluation problem for* ECRPQ($\sqsubseteq$) *queries is decidable.*

Of course the complexity is extremely high as we already know from Corollary V.8.

Note that while the intersection problem of $\sqsubseteq$ with rational relations is decidable, as is $\text{GENINT}_{\sqsubseteq}(\text{REG})$, we lose the decidability of $\text{GENINT}_{\sqsubseteq}(\text{RAT})$ even in the simplest cases that go beyond the intersection problem (that is, for ternary relations in RAT and any I that does not force two words to be the same).

Proposition VI.3. *The problem $(\text{RAT} \cap_I \sqsubseteq) \stackrel{?}{=} \emptyset$ is undecidable even over ternary relations when I is one of the following: $\{(1,2),(2,3)\}$, or $\{(1,2),(1,3)\}$, or $\{(1,2),(3,2)\}$.*

B. Generalized intersection problem for recognizable relations

We now consider the problem of answering CRPQs with rational relations S, or, equivalently, the problem $\text{GENINT}_S(\text{REC})$. Recall that an instance of such a problem consists of an m-ary recognizable relation R and a set $I \subseteq [m]^2$. The question is whether $R \cap_I S \neq \emptyset$, i.e., whether there exists a tuple $(w_1, \ldots, w_m) \in R$ so that $(w_i, w_j) \in S$ whenever $(i,j) \in I$. It turns out that the decidability of this problem hinges on the graph-theoretic properties of I. In fact we shall present a *dichotomy result*, classifying problems $\text{GENINT}_S(\text{REC})$ into PSPACE-complete and undecidable depending on the structure of I.

Before stating the result, we need to decide how to represent a recognizable relation R. Recall that an m-ary $R \in \text{REC}$ is a union of relations of the form $L_1 \times \ldots \times L_m$, where each L_i is a regular language. Hence, as the representation of R we take the set of all such L_is involved, and as the measure of its complexity, the total size of NFAs defining the L_is.

With a set $I \subseteq [m]^2$ we associate an *undirected* graph G_I whose nodes are $1, \ldots, m$ and whose edges are $\{i,j\}$ such that either $(i,j) \in I$ or $(j,i) \in I$. We call an instance of $(\text{REC} \cap_I S) \stackrel{?}{=} \emptyset$ *acyclic* if G_I is an acyclic graph.

Now we can state the dichotomy result.

Theorem VI.4.

- *Let S be a binary rational relation. Then acyclic instances of $\text{GENINT}_S(\text{REC})$ are decidable in* PSPACE*. Moreover, there is a fixed binary relation S_0 such that the problem $(\text{REC} \cap_I S_0) \stackrel{?}{=} \emptyset$ is* PSPACE*-complete.*
- *For every I such that G_I is not acyclic, there exists a binary rational relation S such that the problem $(\text{REC} \cap_I S) \stackrel{?}{=} \emptyset$ is undecidable.*

Proof idea. For PSPACE-hardness we can do an easy reduction from nonemptiness of NFA intersection. Given m NFAs $\mathcal{A}_1, \ldots, \mathcal{A}_m$, define the (acyclic) relation $I = \{(i, i+1) \mid 1 \leq i < m\}$. Then $\bigcap_i \mathcal{L}(\mathcal{A}_i)$ is nonempty iff $\prod_i \mathcal{L}(\mathcal{A}_i) \cap_I S_0 \neq \emptyset$, where $S_0 = \{(w,w) \mid w \in \Sigma^*\}$.

For the upper bound, we show how to construct, in exponential time, for each m-ary recognizable relation R, a binary rational relation S and an acyclic $I \subseteq [m]^2$, an m-ary transducer $\mathcal{A}(R,S,I)$ that accepts precisely those $\bar{w} = (w_1,\ldots,w_m) \in (\Sigma^*)^m$ such that $\bar{w} \in R$ and $(w_i,w_j) \in S$, for each $(i,j) \in I$. Intuitively, $\mathcal{A}(R,S,I)$ represents the "synchronization" of the transducer that accepts R with a copy of the transducer that recognizes S over each projection defined by the pairs in I. Such synchronization is possible since I is acyclic. Hence, in order to solve $\text{GENINT}_S(\mathsf{REC})$ we only need to check $\mathcal{A}(R,S,I)$ for nonemptiness. The latter can be done in PSPACE by the standard "on-the-fly" reachability analysis. □

C. CRPQs with rational relations

The acyclicity condition gives us a robust class of queries, with an easy syntactic definition, that can be extended with *arbitrary* rational relations. Recall that CRPQ(S) queries are those of the form

$$\varphi(\bar{x}) \;=\; \exists \bar{y}\,\Big(\bigwedge_{i=1}^{m}(u_i \xrightarrow{\chi_i:L_i} u_i') \;\wedge\; \bigwedge_{(i,j)\in I} S(\chi_i,\chi_j)\Big),$$

see (2) in Sec.IV. We call such a query *acyclic* if G_I, the underlying undirected graph of I, is acyclic.

Theorem VI.5. *The query evaluation problem for acyclic* CRPQ*(S) queries is decidable for every binary rational relation S. Its combined complexity is* PSPACE*-complete, and data complexity is* NLOGSPACE*-complete.*

Thus, we get not only the possibility of extending CRPQs with rational relations but also a good complexity of query evaluation. The NLOGSPACE-data complexity matches that of RPQs, CRPQs, and ECRPQs [15], [16], [3], and the combined complexity matches that of first-order logic, or ECRPQs without extra relations.

The next natural question is whether we can recover decidability for weaker syntactic conditions by putting restrictions on a class of relations S. The answer to this is positive if we consider *directed* acyclicity of I, rather than acyclicity of the underlying undirected graph of I. Then we get decidability for the class of SCR relations. In fact, we have a dichotomy similar to that of Theorem VI.4.

Theorem VI.6.

- *Let S be a relation from* SCR*. Then* $(\mathsf{REC} \cap_I S) \stackrel{?}{=} \emptyset$ *is decidable in* NEXPTIME *if I is a directed acyclic graph.*
- *There is a relation I with a directed cycle and $S \in \mathsf{SCR}$ such that* $(\mathsf{REC} \cap_I S) \stackrel{?}{=} \emptyset$ *is undecidable.*

In particular, if we have a CRPQ(S) query (2) where I is acyclic (as a directed graph) and $S \in \mathsf{SCR}$, then query evaluation has NEXPTIME combined complexity.

The proof of this result is quite different from the upper bound proof of Theorem VI.4, since the set of witnesses for the generalized intersection problem is no longer guaranteed to be rational without the undirected acyclicity condition. Instead, here we establish the finite-model property, which implies the result.

Also, as a corollary to the proof of Theorem VI.6, we get the following result:

Proposition VI.7. *Let $S \in \mathsf{SCR}$ be a partial order. Then* $\text{GENINT}_S(\mathsf{REC})$ *is decidable in* NEXPTIME*.*

Corollary VI.8. *If $S \in \mathsf{SCR}$ is a partial order, then* CRPQ*(S) queries can be evaluated with* NEXPTIME *combined complexity. In particular,* CRPQ*($\sqsubseteq$) queries have* NEXPTIME *combined complexity.*

The last question is whether these results can be extended to other relations considered here, such as subword and suffix. We do not know the result for subword (which appears to be hard), but we do have a matching complexity bound for the suffix relation.

Proposition VI.9. *The problem* $\text{GENINT}_{\preceq_{\text{suff}}}(\mathsf{REC})$ *is decidable in* NEXPTIME*. In particular,* CRPQ*($\preceq_{\text{suff}}$) queries can be evaluated with* NEXPTIME *combined complexity.*

VII. CONCLUSIONS

Motivated by problems arising in studying logics on graphs (as well as some verification problems), we studied the intersection problem for rational relations with recognizable and regular relations over words. We have looked at rational relations such as subword $\preceq$, suffix $\preceq_{\text{suff}}$, and subsequence $\sqsubseteq$, which are often needed in graph querying tasks. The main results on the complexity of the intersection and generalized intersection problems, as well as the combined complexity of evaluating different classes of logical queries over graphs are summarized in Fig. 1. Several results generalizing those (e.g., to the class of SCR relations) were also shown. Two problems related to the interaction of the subword relation with recognizable relations remain open and appear to be hard.

From the practical point of view, as rational-relation comparisons are demanded by many applications of graph data, our results essentially say that such comparisons should not be used together with regular-relation comparisons, and that they need to form acyclic patterns (easily enforced syntactically) for efficient evaluation.

So far we dealt with the classical setting of graph data [1], [8], [9], [15], [16] in which the model of data is that of a graph with labels from a finite alphabet. In both graph data and verification problems it is often necessary to deal with the extended case of infinite alphabets (say, with graphs holding data values describing its nodes), and languages that query both topology and data have been proposed recently [23], [25]. A natural question is to extend the positive results shown here to such a setting.

Acknowledgments: We thank Sylvain Schmitz for helpful comments and suggestions. Partial support provided by Fondecyt grant 1110171, EPSRC grant G049165, and FET-Open Project FoX, grant agreement 233599. Part of this work was done when the first author visited Edinburgh, and the third author visited Santiago.

	$R \in \mathsf{REC}$	$R \in \mathsf{REG}$	$R \in \mathsf{RAT}$
$(R \cap \preceq) \stackrel{?}{=} \emptyset$	PTIME (cf. [5])	undecidable	undecidable
$(R \cap \preceq_{\mathrm{suff}}) \stackrel{?}{=} \emptyset$		undecidable	undecidable
$(R \cap \sqsubseteq) \stackrel{?}{=} \emptyset$		decidable, non-multiply-recursive	decidable, non-multiply-recursive [12]
$(R \cap_I \preceq) \stackrel{?}{=} \emptyset$	?	undecidable	undecidable
$(R \cap_I \preceq_{\mathrm{suff}}) \stackrel{?}{=} \emptyset$	NEXPTIME	undecidable	
$(R \cap_I \sqsubseteq) \stackrel{?}{=} \emptyset$	NEXPTIME	decidable, non-multiply-recursive	

	$S = \sqsubseteq$	$S = \preceq_{\mathrm{suff}}$	$S = \preceq$	S arbitrary in RAT
ECRPQ(S)	decidable, non-multiply-recursive	undecidable	undecidable	undecidable
CRPQ(S)	NEXPTIME	NEXPTIME	?	undecidable
acyclic CRPQ(S)	PSPACE	PSPACE	PSPACE	PSPACE

Fig. 1. Complexity of the intersection and generalized intersection problems, and combined complexity of graph queries for subword ($\preceq$), suffix ($\preceq_{\mathrm{suff}}$), and subsequence ($\sqsubseteq$) relations

REFERENCES

[1] R. Angles, C. Gutiérrez. Survey of graph database models. *ACM Comput. Surv.* 40(1): (2008).

[2] K. Anyanwu, A. P. Sheth. ρ-Queries: enabling querying for semantic associations on the semantic web. In *WWW'03*.

[3] P. Barceló, C. Hurtado, L. Libkin, P. Wood. Expressive languages for path queries over graph-structured data. In *PODS*, pages 3-14, 2010.

[4] M. Benedikt, L. Libkin, T. Schwentick, L. Segoufin. Definable relations and first-order query languages over strings. *J. ACM* 50(5):694-751 (2003).

[5] J. Berstel. *Transductions and Context-Free Languages*. B. G. Teubner, 1979.

[6] A. Blumensath and E. Grädel. Automatic structures. In *LICS'00*, pages 51–62.

[7] V. Bruyère, G. Hansel, C. Michaux, R. Villemaire. Logic and p-recognizable sets of integers. *Bull. Belg. Math. Soc.* 1 (1994), 191–238.

[8] D. Calvanese, G. de Giacomo, M. Lenzerini, M. Y. Vardi. Containment of conjunctive regular path queries with inverse. In *KR'00*, pages 176–185.

[9] D. Calvanese, G. de Giacomo, M. Lenzerini, M. Y. Vardi. View-based query processing and constraint satisfaction. In *LICS*, pages 361-371, 2000.

[10] L. Cardelli, P. Gardner, G. Ghelli. A spatial logic for querying graphs. In *ICALP'02*, pages 597-610.

[11] O. Carton, C. Choffrut, S. Grigorieff. Decision problems among the main subfamilies of rational relations. *RAIRO/ITA*, 40 (2006), 255–275.

[12] P. Chambart, Ph. Schnoebelen. Post embedding problem is not primitive recursive, with applications to channel systems. In *FSTTCS'07*, pages 265–276.

[13] P. Chambart, Ph. Schnoebelen. The ordinal recursive complexity of lossy channel systems. *LICS'08*, pages 205–216.

[14] C. Choffrut. Relations over words and logic: a chronology. *Bulletin of the EATCS* 89 (2006), 159–163.

[15] M. P. Consens, A. O. Mendelzon. GraphLog: a visual formalism for real life recursion. In *PODS'90*, pages 404–416.

[16] I. Cruz, A. Mendelzon, P. Wood. A graphical query language supporting recursion. In *SIGMOD'87*, pages 323-330.

[17] A. Dawar, P. Gardner, G. Ghelli. Expressiveness and complexity of graph logic. *Inf.& Comput.* 205 (2007), 263-310.

[18] A. Deutsch, V. Tannen. Optimization properties for classes of conjunctive regular path queries. *DBPL'01*, pages 21–39.

[19] L. E. Dickson. Finiteness of the odd perfect and primitive abundant numbers with n distinct prime factors. *The American Journal of Mathematics*, 35(4):413–422, 1913.

[20] C. Elgot and J. Mezei. On relations defined by generalized finite automata. *IBM J. Res. Develop.* 9 (1965), 47–68.

[21] D. Florescu, A. Levy, D. Suciu. Query containment for conjunctive queries with regular expressions. In *PODS'98*.

[22] C. Frougny and J. Sakarovitch. Synchronized rational relations of finite and infinite words. *TCS* 108 (1993), 45–82.

[23] O. Grumberg, O. Kupferman, S. Sheinvald. Variable automata over infinite alphabets. In *LATA'10*, pages 561–572.

[24] G. Higman. Ordering by divisibility in abstract algebras. *Proc. London Math. Soc. (3)*, 2(7):326–336, 1952.

[25] L. Libkin, D. Vrgoč. Regular path queries on graphs with data. In *ICDT'12*.

[26] L. Lisovik. The identity problem for regular events over the direct product of free and cyclic semigroups. *Doklady Akad. Nauk Ukr., ser. A*, 6 (1979), 410–413.

[27] M.H. Löb and S.S. Wainer. Hierarchies of number theoretic functions, I. *Arch. Math. Logik Grund.*, 13:39–51, 1970.

[28] M. Nivat. Transduction des langages de Chomsky. *Ann. Inst. Fourier* 18 (1968), 339–455.

[29] H. Rose. *Subrecursion: Functions and Hierarchies*. Clarendon Press, 1984.

[30] W. Thomas. Infinite trees and automaton-definable relations over ω-words. *TCS* 103 (1992), 143–159.